Control Flow: Conditional Statements

The ability to select what actions to perform based on circumstances we encounter is critical in real life and also in programming. A **conditional statement** (also called a **selection statement**) allows a program to do that:

- An *if-statement* instructs the shell to select between executing a suite or just skipping the suite.
- An *if-else statement* instructs the shell to select exactly one of two suites to execute.
- An *if-elif statement* or an *if-elif-else statement* instructs the shell to select exactly one of two or more suites to execute.

This exercise focuses on using *if-else* statements to control the flow of execution.

**Part (a):** With a partner, play the game “What kind of animal were you in a prior lifetime?” (One of you plays the Host role and the other plays the Guest Player role; then switch roles.) After you have played it a couple of times, we will demo a Python program that automates the game.

**Part (b):** With your partner, create a folder, say *Week3*, for this week's exercises. Download *game1.py* into this folder and then load it into Spyder. Execute it a few times. Study it and be ready to answer the following questions about it.

1. Where does the first *if-else* statement start and where does it end?

2. Where does the second *if-else* statement start and where does it end?

3. What lines of code are executed if the user types
   
   a. *Y* at the first prompt and *n* at the second?
   
   b. *y* at the first prompt and *y* at the second?
   
   c. *n* at the first prompt?

4. In what sense have you thoroughly tested this program?
**Part (c):** The `print('NOT DONE')` statement in this program is called a **print stub**. A programmer uses print stubs in order that a program can be developed **incrementally**. The print stubs allow a partial implementation to be run and tested. The programmer then replaces the print stubs with code, one after the other, testing each new version of the program thoroughly before proceeding further.

To give you practice incrementally writing a program, finish implementing this program, as follows:

1. Replace the print stub with code that implements the True (YES) branch of the next question (Do you like cheese?) and uses a print stub on the False (NO) branch. **Run two tests of the new program before proceeding** - one that causes the program to print *You were a mouse!* and one that causes it to print *NOT DONE*.

2. Replace the print stub with code that implements the last question (Do you like to swim?). Run two tests of the new program - one that prints *You were a fish!* and one that prints *You were a cat!*

3. What advantages come from implementing the program in this fashion instead writing and testing the full program in one sitting?

**Part (d):** [Extra for experts] If time permits, add error checking to your program in part (c). In other words, modify the program so that it plays the game as shown in the diagram provided that the user enters Y, y, N, or n at each prompt; but prints an error message and quits, if the user types any other input at any of the prompts. (Hint: The *if-elif-else* statement will be useful.)